⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　昭62－295174

⑤1Int.Cl.⁴	識別記号	庁内整理番号	④3公開　昭和62年(1987)12月22日
G 06 F 15/332 15/16		A－8320－5B T－2116－5B	

審査請求　有　発明の数 2　（全13頁）

⑤4発明の名称　並列データ処理装置

⑳特　願　昭61－137313

㉒出　願　昭61(1986)6月14日

⑦2発明者　宮田　裕行　鎌倉市大船5丁目1番1号　三菱電気株式会社情報電子研究所内

⑦1出願人　工業技術院長

明　細　書

1．発明の名称

並列データ処理装置

2．特許請求の範囲

(1)加減算および乗算機能を有する演算手段と記憶手段と与えられた命令の実行を制御する実行制御手段とを夫々有する複数の基本演算要素が、互いに隣接する要素間でデータを転送するためのデータ転送ラインと外部からのデータを要素間を順次経由して入力するとともに外部にデータを出力するためのデータ入出力ラインと要素の行単位あるいは列単位に同一の値を外部から与えるためのブロードキャストラインとにより2次元状に相互接続された演算アレイと、上記2次元状に接続された各基本演算要素内に1つの処理対象データあるいは複数個の処理対象データを対応させ、これらのデータに対し行単位および列単位に順に2のべき乗だけ離れた要素間でデータを転送するとともに各々の要素内での加減算，乗算を制御する制御手段とを備え、上記各演算を繰り返すことにより、所定のアルゴリズムによる高速フーリエ変換のバタフライ演算を並列に実行することを特徴とする並列データ処理装置。

(2)加減算および乗算機能を有する演算手段と記憶手段と与えられた命令の実行を制御する実行制御手段とを夫々有する複数の基本演算要素が、互いに隣接する要素間でデータを転送するための、データ転送ラインと外部からのデータを要素間を順次経由して入力するとともに外部にデータを出力するためのデータ入出力ラインと要素の行単位あるいは列単位に同一の値を外部から与えるためのブロードキャストラインとにより2次元状に相互接続された演算アレイと、上記2次元状に接続された各基本演算要素内に1つの処理対象データあるいは複数個の処理対象データを対応させ、これらのデータに対し行単位および列単位に順に2のべき乗だけ離れた要素間でデータを転送するとともに各々の要素内での加減算，乗算を制御し、かつ上記各演算の終了後に各要素内のデータを所定の行単位および列単位に入れ換え処理する制御

手段とを備え、上記各演算を繰り返すことにより、所定のアルゴリズムによる高速フーリエ変換のバタフライ演算を並列に実行するとともに、上記入れ換え処理を繰り返すことにより、高速フーリエ交換のビットリバース処理を実現することを特徴とする並列データ処理装置。

3．発明の詳細な説明

〔産業上の利用分野〕

この発明は、高速フーリエ変換を高速に実行するために、該変換のバタフライ演算を並列に実行する並列データ処理装置に関するものである。

〔従来の技術〕

まず、この発明における高速フーリエ変換について簡単に説明する。高速フーリエ変換とは、次に示す離散的フーリエ変換式

$$F(n)=\frac{1}{N}\sum_{k=0}^{N-1}f(k)W^{nk}$$

$$\left(W=e^{-\frac{2\pi}{N}j}\right)$$

を高速に求めるもので、サンプル値f（k）と回転子W^{nk}の乗算および、そのΣに対する加減算回数を大幅に減少させたものである。

この高速フーリエ変換のアルゴリズムは種々提案されているが、本発明においては以後に示すインプレイス型と呼ばれるものを使用する。ここでは便宜上、高速フーリエ変換の基本となるバタフライ演算の記述を第3図のように行う。この第3図の記述方法により

$X=x+y$

$Y=(x-y)W^{k}\qquad\left(W=e^{-\frac{2\pi}{N}j}\right)$

を表わすものとする。この記述方法を使用すると、本発明で用いるアルゴリズムは第4図に示す様になる。ただし、この図は、16ポイントの高速フーリエ変換を表わす。このアルゴリズムの特徴は、入力されたサンプルデータが順に2のべき乗ポイントだけ離れたデータとバタフライ演算を行い、その結果を元の2つのデータと同じ位置に格納する点である。また、高速フーリエ変換後のデータをf（i）とすると、これらはiを後に述べるビットリバース処理を施した順に並ぶ。以後では、説明を簡単にするために、16ポイントの高速フーリエ変換について説明するが、ポイント数が増加しても（ただしポイント数は2のべき乗）同様に扱える。また、説明上、第4図に示す様に各バタフライ演算を左から順に、第1ステージ14，第2ステージ15，第3ステージ16，第4ステージ17と名付ける。ちなみに、nポイント（nは2のべき乗）の高速フーリエ変換は、第$\log_2 n$ステージまで存在する。また、nポイントの高速フーリエ変換において、第1ステージでは、各データがn／2ポイント離れたデータとの間でバタフライ演算を行う。第2ステージは、データを連続したn／2ポイントずつの2つのブロックに分離し、各ブロック内でn／4ポイント離れたデータとの間でバタフライ演算を行う。以下同様であり、第iステージでは、データを連続したn／2^{i-1}ポイントずつの2^{i-1}個のブロックに分離し、各ブロック内でn／2^{i}ポイント離れたデータとの間でバタフライ演算を行うことになる。

さて、従来は、この高速フーリエ変換を汎用計算機を用いて逐次的に行うか、第28図に示す加算器47，減算器48，乗算器49を組み合わせたバタフライ演算器46を繰り返し使用するか、複数個のバタフライ演算器46を使用して実現していた。

次に動作について説明する。ここでは、第4図の16ポイントの高速フーリエ変換を行う場合を考える。1つのバタフライ演算器46のみを使用する場合には、まず第1ステージ14の個々の8組のペア（f（0）とf（8），f（1）とf（9）など）とその対応する回転子W^{k}を順にバタフライ演算器46に入力する。以下、各演算結果を使用して第2，3，4ステージのバタフライ演算を各々順に行う。各ステージごとに8回のバタフライ演算を行えばよいため、全体で32回のバタフライ演算を行うことにより処理は終了する。

Nポイントの高速フーリエ変換の場合には、$\frac{N}{2}\log_2 N$（但しNは2のべき乗）回のバタフラ

イ演算を行えばよいことになる。ただし、バタフライ演算器46へ入力するためのデータをメモリから読み出したり、演算結果をメモリに書き込むためのアドレス計算は毎回必要となる。

更に高速化を考えた場合には、同一ステージ内のバタフライ演算を並行して行う方法があげられる。例えば、第4図においては、8個のバタフライ演算器46を用意する。8個のバタフライ演算器46を1回使用することにより各ステージの処理は一度に終了する。そのため、全体としてステージの数だけバタフライ演算器46を使用することにより、高速フーリエ変換が行える。ちなみに、Nポイントの高速フーリエ変換の場合には、N／2個のバタフライ演算器46を用意して$\log_2 N$回（但しNは2のべき乗）使用することにより、処理が終了する。ただし、先に示した各データに対するアドレス計算に加えて、全データを毎回メモリから読み出したり、メモリに書き込む処理は必要となる。

〔発明が解決しようとする問題点〕

以上述べた、複数個のバタフライ演算器46を使用する方式では、実際的には次に示す点から高速な処理が可能ではない。

1）処理対象であるサンプル値f（k）の数が非常に多い場合、最も高速に高速フーリエ変換を行うには多数のバタフライ演算器46が必要となり、現実的ではない。そのため、実際には少数のバタフライ演算器46を繰り返し使用することになる。

2）各ステージでバタフライ演算器46の数だけ並列に実行できるが、各バタフライ演算器46に入力するデータおよび出力するデータは通常メモリに格納されており、各データごとにアドレス計算が必要となる。このアドレス計算およびメモリからのロード，ストアは逐次的に行わざるを得ず、演算器の並列性が生かしきれない。

従来のバタフライ演算器46を用いた方式では、高速フーリエ変換の処理アルゴリズムによらず上記の欠点が存在するため、必ずしも高速な処理が望めないという問題点があった。

この発明は上記のような問題点を解消するためになされたもので、高速に高速フーリエ変換を実行することができる並列データ処理装置を得ることを目的とする。

〔問題点を解決するための手段〕

この発明に係る並列データ処理装置は、加減算および乗算機能を有する演算手段と記憶手段と与えられた命令の実行を制御する実行制御手段とを夫々有する複数の基本演算要素が、互いに隣接する要素間でデータを転送するためのデータ転送ラインと外部からのデータを要素間を順次経由して入力するとともに外部にデータを出力するためのデータ入出力ラインと要素の行単位あるいは列単位に同一の値を外部から与えるためのブロードキャストラインとにより2次元状に相互接続された演算アレイと、上記2次元状に接続された各基本演算要素内に1つの処理対象データあるいは複数個の処理対象データを対応させ、これらのデータに対し行単位および列単位に順に2のべき乗だけ離れた要素間でデータを転送するとともに各々の要素内での加減算，乗算を制御する制御手段とを備えたものである。

また、この発明の別発明は、上記演算アレイと、上記2次元状に接続された各基本演算要素内に1つの処理対象データあるいは複数個の処理対象データを対応させ、これらのデータに対し行単位および列単位に順に2のべき乗だけ離れた要素間でデータを転送するとともに各々の要素内での加減算，乗算を制御し、かつ上記演算の終了後に各要素内のデータを所定の行単位および列単位に入れ換え処理する制御手段とを備えたものである。

〔作用〕

この発明における並列データ処理装置では、高速フーリエ変換におけるバタフライ演算で必要となるデータが、基本演算要素間で並列に転送され、各要素においてバタフライ演算が同時に並列して行えるため、従来のメモリアドレス計算，メモリ競合の問題を解消でき、高速に高速フーリエ変換が実行できる。

また、この発明の別発明においては、上記演算

後、所定の行単位および列単位に入れ換え処理を繰り返すことにより、高速フーリエ変換のビットリバース処理を実現することができるので、更に、高速に高速フーリエ変換が実行できる。

〔実施例〕

以下、この発明の一実施例を図について説明する。第1図において、1はこの発明における並列データ処理装置を構成する基本演算要素（以下PEと記す）、2はPE1内のデータを格納する記憶手段としてのメモリ、3は各PE1において加算，減算，乗算を実行する演算手段としてのALU、4は各PE1内での演算命令等を修飾するための実行制御手段としての制御フラグ、5，6，7，8は各々北，東，西，南方向（図中上，右，左，下方向）に存在するPE1とのデータ転送に使用するデータ転送ライン、9，10は各々、後述する演算アレイの行単位，列単位に、外部より送られた同一の値を各PE1に転送するためのブロードキャストライン、11，12は外部から各PE1を順に経由してデータの入力を行う時、あるいは外部へ同様にしてデータを出力する時に使用するデータ入出力ラインである。

第2図は、本発明の一実施例であり、13aは第1図に示したPE1を上記各ラインを介して2次元状に4×4個組み合わせた場合の演算アレイ、13bは上記演算アレイ13aにおける各PE1間のデータの転送および各PE1内での演算の実行を制御するマイクロプロセッサ等からなる制御手段である。

以下、本発明における実施例の作用，動作の詳細な説明を行う。簡単のため、256ポイントに対する高速フーリエ変換を、16×16個のPEから構成される並列データ処理装置により実行する場合を例にとって説明する。

なお、以下の説明のために使用する記号を定義する。

ここでは、

$$F(n)=\frac{1}{256}\sum_{k=0}^{255} f(k)\,W^{nk}$$

$$\left(W=e^{-\frac{2\pi}{256}j}\right)$$

$$0\leq n\leq 255$$

により表わされる高速フーリエ変換を扱う。f（i）が入力データ，F（i）が出力データである。また、第5図に16×16個のPEから構成される並列データ処理装置の表記例18を示す。並列データ処理装置を構成するPEを各行単位に、順に上から第0行，第1行，…，第15行と呼び、各列単位に順に左から第0列，第1列，…，第15列と呼ぶ。また、特定のPEを示す場合には、第i行第j列のPEをPE（i，j）と呼ぶことにする。また第i－j行（列）のPEという表現で第i行（列）から第j行（列）までのPEを表わす。

まず並列データ処理装置へのデータ入力方法について述べる。対象とするデータ数は256個であり、PEの数も256個であるため1PEに1個のデータを対応させる。具体的には第6図19に示す様になる。すなわち、PE（0，0）にf（0）を格納し、以下列方向に順に格納する。PE（15，0）にf（15）が格納された後、第2列のPEに同様に順に格納していく。以下同様であり、最後のf（255）はPE（15，15）に格納される。換言すると、PE（i，j）に収められるデータはf（i＋16j）である。逆に、f（x）（x＝〔x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0），（　）は2進数表現を表わす）はi＝〔x_3 x_2 x_1 x_0〕，j＝〔x_7 x_6 x_5 x_4〕なるPE（i，j），すなわち第i行第j列のPEに格納される。

データの入力後、先に第4図の16ポイントの高速フーリエ変換の例で示した様に各ステージごとにバタフライ演算を行う。この第4図で示した様に、256ポイントの高速フーリエ変換の第1ステージでは、各々128ポイント離れたデータどうしのバタフライ演算を行えばよい。

今、第6図に示す様に各PEに入力されたデータで考える。すると、128ポイント離れた2つのデータは、各々同一の行のPE内に存在し、か

つ、その各々のＰＥはちょうど８ＰＥ分離れている。これはＰＥ（ｉ，ｊ）内のデータがｆ（ｉ＋１６ｊ）となる点からも明らかである。言い換えれば、並列データ処理装置内の２次元データを第７列と第８列の境目で左右に２分割し、その一方を片方の上に一致する様にずらした時、各対応するデータがバタフライ演算を行うデータのペアとなっている。そのため、第７図（ｂ）に示す様に、データ２０を並列データ処理装置２１上で西方向（図では左方向）へ８ＰＥ分移動し、並列データ処理装置２１の第０－７列のＰＥがその移動されたデータを取り込むことにより、これらのＰＥ内に対応するデータどうしが格納されることになる。（なお、左半分のＰＥだけがデータを取り込める操作は、第１図ＰＥ（１）内の制御フラグ４の制御による。）ここで述べたデータ移動の具体的な処理結果を示すと第８図２２となる。

第８図の結果を用いて、並列データ処理装置の第０～７列のＰＥだけでバタフライ演算を行えば第１ステージの処理は終了する。ところが、これでは全ＰＥの稼動率は１／２となり、効率が悪い。そこで、次の様にする。

もともとバタフライ演算は２つの同一データの和と差を求めるものであるため、この和を第０－７列のＰＥで、差を残りの第８－１５列のＰＥで同時に求めることにする。そのため、まず、第８－１５列のＰＥにも同一のデータを格納する必要がある。第８－１５列のＰＥには既にｆ（１２６）～ｆ（２５５）のデータが入力されているため、今度は第０－７列の各ＰＥ内のデータを第８－１５列のＰＥに移動する。これを第７図（ｃ）に示す。データ２０を８ＰＥ分右方向へシフトし、並列データ処理装置２１の第８－１５列のＰＥだけが、制御フラグの制御のもとシフトされたデータを取り込む。ここに述べたデータ移動の具体的な処理結果を示すと第９図２３となる。

以上の様に、並列データ処理装置内でデータのシフトを終えた後は、並列データ処理装置の第０－７列のすべてのＰＥで加算を、第８－１５列のすべてのＰＥで減算を同時に行う。ＰＥにより行う演算を制御するのは各ＰＥ内の制御フラグによる。すなわち、各ＰＥの存在する位置により、加算か減算かを決定し制御フラグで操作する。

加算および減算の処理が終了した後は、バタフライ演算の項で示した様に、減算した値に回転子をかけ合わせる処理が必要である。減算した値は、第８－１５列のＰＥに格納されているため、これらのＰＥに対応する回転子を入力し、積を求める。

結果のデータの格納先は、ｆ（ｉ）とｆ（ｉ＋１２８）（$0 \leq i \leq 127$）のバタフライ演算では加算値がｆ（ｉ），減算かつ回転子との乗算値がｆ（ｉ＋１２８）の位置であるが、上記の方法では加算をｆ（ｉ）の位置のＰＥで、減算かつ乗算をｆ（ｉ＋１２８）の位置のＰＥで行っているため結果のデータを入れ換える必要はない。

さて、以上で第１ステージに対する処理は終了した。次に、第２ステージの処理に移るわけであるが、同様の手法に基づいて行うことができる。

すなわち第２ステージでは、２５６ポイントのデータを前半と後半の１２８ポイントずつのブロックに分け、各ブロック内では６４ポイント離れたデータどうしでバタフライ演算を行えばよい。並列データ処理装置内で考えると、第０－３列のＰＥと第４－７列のＰＥ，第８－１１列のＰＥと第１２－１５列のＰＥどうしで対応するＰＥがバタフライ演算を行うデータのペアを保持している。今、第１ステージの処理前のデータ配置を第１０図（ａ）で示す様に表わす。すなわち、斜線を施した部分に存在するＰＥと白ぬきの部分に存在するＰＥどうしでデータの交換を行い、斜線部で加算を白ぬき部で減算および乗算を行った。同様の記述を用いると、第２ステージは第１０図（ｂ）に示すようになる。すなわち、並列データ処理装置を第０－７列のＰＥと第８－１５列のＰＥで夫々等分し、各ブロック内で第１ステージと同様の処理を行えばよい。すなわち第０－３列のＰＥと第４－７列のＰＥ間でデータの交換を行い、第０－３列のＰＥで加算，第４－７列のＰＥで減算および乗算を行う。第８－１１列のＰＥと第１２－１５列のＰＥとの間の処理も同じである。

以下、第3ステージ，第4ステージでの処理も各々第10図（c），（d）に示す様に並列データ処理装置内のデータを列方向に4分割，8分割し、各ブロック内で第1ステージと同様の処理を施すことにより各ステージでのバタフライ演算が行える。

さて、第4ステージの場合は、各ブロックが2列のＰＥから成り立っており、この2列のＰＥ内に存在するデータ間でバタフライ演算が行われる。この時の2つのデータは16ポイント離れている。

この次の第5ステージを考えた場合、これ以上列単位のブロック分割は行えない。バタフライ演算を行う2つのデータは8ポイント離れていることになる。すなわち、第6図からもわかる様に、第5ステージ以降のバタフライ演算を考えた場合、対応するデータどうしは、今度は行方向に2分割，4分割して対応していく。例えば第5ステージの対応を先の第10図の要領で表わせば第11図（a）となる。

すなわち、第7図と同様に、第12図で示す様にデータ24を並列データ処理装置25上で北方向（図中上方向）へ転送し、第0－7行のＰＥでデータを取り込み、次にデータ24を南方向（図中下方向）へ転送し、第8－15行のＰＥでデータを取り込むことによりバタフライ演算を行うべきデータが各ＰＥにそろえられる。今、第4ステージの処理が終了した各データをf（i）（0≦i≦255）で表わすと、第12図（a）の処理後の各ＰＥ内データは第13図26、第12図（b）の処理後の各ＰＥ内データは、第14図27となる。この後、第0－7行のＰＥで加算を，第8－15行のＰＥで減算および乗算を行うことにより第5ステージの処理は終了する。

以下、第6ステージ，第7ステージ，第8ステージは各々第2ステージ，第3ステージ，第4ステージの列方向に行っていた処理を行方向に変更しただけで、まったく同様の処理を行えばよい。これらを第10図（b），（c），（d）と同じく、各々、第11図（b），（c），（d）に示す。最後の第8ステージ，すなわち第11図（d）は1ポイントだけ離れたデータどうしのバタフライ演算であり、この処理により全バタフライ演算は終了する。

さて、これまでに示した第1ステージ～第8ステージの処理により、高速フーリエ変換におけるバタフライ演算は終了するが、第4図の16ポイントの高速フーリエ変換の例で示した様に、得られる値F（i）はiに対して昇べきの順ではなく、iを後に述べるビットリバース処理した順に並んでいる。そのため、次には、この順序を並べ変える操作が必要となる。この処理は、外部のメモリ上で行うこともできるが、データの数だけの処理時間がかかるため、この並列データ処理装置上で行うことにする。

まずは、このビットリバース処理を第4図と同様に16ポイントのビットリバース処理を用いて説明する。

第15図を用いる。今、f（0）からf（15）までの16個の値が与えられたとする。これらの値の各インデックス値0～15を2進数で表わす。これを〔a_3 a_2 a_1 a_0〕（以後、文章中，図中において〔x〕はxが2進数表現であることを示す）とすると、この各ビットを〔a_0 a_1 a_2 a_3〕と、前後で全体を入れ換えることをビットリバースと呼ぶ。一般には、〔b_n b_{n-1} … b_1 b_0〕を〔b_0 b_1 … b_{n-1} b_n〕と変換する処理を示す。

ビットリバース処理とは、先に述べた16ポイントの例ではf（0）からf（15）の値を各インデックス値のビットリバースした値をインデックスとするfの値と入れ換えることである。f（0）からf（15）に対してビットリバース処理を施した例を第15図28に示す。

ある値にビットリバース処理したものにもう1度ビットリバース処理を行うと、もとの値に戻る。すなわち、第15図においてビットリバース処理が施された値をもととして、もう1度ビットリバース処理を行うともとのf（0）～f（15）に戻る。すなわち、先に示したバタフライ演算の結果、各ＰＥ内にはデータがビットリバース処理を

施した順に格納されているため、ここにビットリバース処理を施すことにより、もとの昇順に戻ることになる。

第16図29にバタフライ演算後の各PE内に存在するf（j）の値を示す。各々、最初のサンプルデータf（i）をiについてビットリバースした値jをインデックスとした値となっている。

さて、第16図29バタフライ演算の結果を昇順に並べ直すわけであるが、ここでは、もとの第6図19に示す列方向に昇順ではなく、第17図30に示す行方向に昇順となる様に並べ直すものとする。すなわち、第6図19ではPE（i，j）にf（i＋16j）が格納されていたが、出力時には、第17図30に示す様にF（16i＋j）が格納されていればよい。換言すると、入力時の第6図19では、f（x）（x＝〔x7 x6 x5 x4 x3 x2 x1 x0〕）はi＝〔x3 x2 x1 x0〕，j＝〔x7 x6 x5 x4〕なるPE（i，j），すなわち第i行第j列のPEに格納されたが、出力時の第17図30では、F（x）（x＝〔x7 x6 x5 x4 x3 x2 x1 x0〕）はPE（j，i）に格納すればよいことになる（行列で言えば入力データのインデックス値の転置行列したものが出力データのインデックス値となる）。

次に、このビットリバース処理の説明を行う。第18図を使用する。最初に、各PEは第6図19に示した列方向の順にデータを格納している。この後、バタフライ演算によるビットリバース処理のためデータがどのように移動するかを考察する。インデックスが〔a7 a6 a5 a4 a3 a2 a1 a0〕であるポイントAに着目する。1PEに1つのデータが格納され、PEが縦に16個、横に16個ずつ存在する並列データ処理装置のため、ポイントAが〔a7 a6 a5 a4 a3 a2 a1 a0〕で表わされるならば、Aを含む1行のPE32には、そのインデックスの2進数表示による下位4ビットが〔a3 a2 a1 a0〕であるデータが格納されている。言い換えると、"x"を0か1の値をとる任意の値として、Aを含む1行のPE32には、そのインデックスが〔x x x x a3 a2 a1 a0〕である値が格納されている。この1行のラインの値へビットリバース処理を施すと〔a0 a1 a2 a3 x x x x〕となり、第18図33に示すライン33になる。つまり、1行のラインはビットリバース処理により1列のラインに変換される。さて、先に、そのインデックスが〔x7 x6 x5 x4 x3 x2 x1 x0〕である値は、出力時には、i＝〔x7 x6 x5 x4〕，j＝〔x3 x2 x1 x0〕なるPE（i，j）に格納すればよかった。すると、Aを含む1行のPE32はそのインデックスが〔x x x x a3 a2 a1 a0〕のため、最終的にはj＝〔a3 a2 a1 a0〕なるPE（i，j），すなわち第〔a3 a2 a1 a0〕列のPE34に格納されればよい。

上記のビットリバース処理ではAを含む1行のPE32は〔a0 a1 a2 a3 x x x x〕，すなわち、第〔a0 a1 a2 a3〕列のPE33に移された。よって、第〔a0 a1 a2 a3〕列のPE33を第〔a3 a2 a1 a0〕列のPEに移動する処理が必要となる。

ところで、第〔a3 a2 a1 a0〕列のPEの値も同じ理由から第〔a0 a1 a2 a3〕のPEに移動することになる。

これらをまとめると、第〔a3 a2 a1 a0〕列のPEと第〔a0 a1 a2 a3〕列のPEを入れ換えればよいことになる。この列の入れ換えは次の手順で行う。

①第〔a3 a2 a1 a0〕列のPEと第〔a0 a2 a1 a3〕列のPEを入れ換える。

②第〔a0 a2 a1 a3〕列のPEと第〔a0 a1 a2 a3〕列のPEを入れ換える。

すなわち、まず最上位ビットと最下位ビットを交換した列との入れ換えを行い、次に最上位から2ビット目と最下位から2ビット目を交換した列との入れ換えを行う。もし、よりビット数が長い場合には同様の処理を中央のビットまで行う。

さて、①の処理の具体的手法を説明する。〔a3 a2 a1 a0〕と〔a0 a2 a1 a3〕は、そのa0とa3の値により、第19図に示す様になる。つまり、a0＝a3＝0，1の時は同じ値を示す

ので入れ換えの必要はない。$a_0=0$, $a_1=1$（$a_0=1$, $a_1=0$）の場合は、各々の列が7の差を持つので、7PE列分離れたPEを入れ換えることになる。具体的な〔$a_2\ a_1\ a_0$〕の値で示すと第20図となる。

この処理は、並列データ処理装置上で容易に実現できる。すなわち、第21図（b）に示す様にデータ35を並列データ処理装置上36で、7PE分右方向へシフトし、該当する各PE（第8列，第10列，第12列，第14列）がそのデータをとりこむ。次に逆方向へ7PE分データ35をシフトし、該当する各PE（第1列，第3列，第5列，第7列）でデータをとりこむ。これにより7PE列離れたPE間での入れ換えが終了する。

次に②の処理について説明する。この処理も①とまったく同様に行える。すなわち第〔$a_0\ a_2\ a_1\ a_3$〕列と第〔$a_0\ a_1\ a_2\ a_3$〕列の入れ換えは、①と同じ考え方で、$a_1=0$, $a_2=1$（$a_1=1$, $a_2=0$）の場合のみ2PE列分離れたPEと入れ換えを行えばよい。第20図と同様に第22図に示す。並列データ処理装置上での処理も、同様に、第21図（d），（e）に示すように2PE列分ずらして行う。

以上示した処理を行うことにより、列方向のビットリバース処理は終了する。

ところで、この列方向のデータの入れ換えを行った時、行方向について調べてみると、同一行の値は必ず同一行に移動している。つまり、列方向の入れ換えの際には、行方向に関しては何の影響も及ぼしていない。これは、行方向の入れ換えについても同じことが言える。すなわち、行方向のPE間のデータ入れ換えと列方向のPE間のデータ入れ換えは、まったく独立に行うことができる。よって、列方向のビットリバース処理後、行方向のビットリバース処理を行えば全体のビットリバース処理が行える。

行方向のビットリバース処理については、行と列が異なるだけで列方向のビットリバース処理とまったく同じである。そのため、第21図と同様に、第23図に示す様に7PE行だけ該当する行のPE内データを入れ換え（第23図（a）（b））、2PE行だけ該当する行のPE内データを入れ換えればよい（第23図（c）（d））。

本発明における各PE内のインデックス値の変遷を16×16個から成るPEの例で示す。第24図に256個のデータを入力した時点のインデックスを示す。

このデータにバタフライ演算を施すことにより、第25図に示すビットリバースのデータが各PEに格納される。その後、ビットリバース処理を施すことにより第26図が得られる。

以上は、256ポイントのデータのビットリバース処理を、16×16PEから成る並列データ処理装置で行う場合を例示したが、これは一般のビットリバース処理についてもあてはまる。

今、2^{2i} ポイントのデータのビットリバース処理を$2^i\times 2^i$ PEから成る並列データ処理装置で行う場合を示す。先の例と同様に、任意の行のデータのインデックスを〔$x\ x\ x\cdots x\ a_{i-1}a_{i-2}\cdots a_0$〕と表わすと、そのビットリバース処理後は〔$a_0\cdots a_{i-2}\ a_{i-1}\ x\cdots x\ x\ x$〕で表わされる列のデータに移される。この列が格納されるべき列のインデックスは〔$a_{i-1}\ a_{i-2}\cdots a_0\ x\cdots x\ x\ x$〕である。また、逆に〔$a_{i-1}\ a_{i-2}\cdots a_0\ x\ x\cdots x\ x\ x$〕の列も〔$a_0\cdots a_{i-2}\ a_{i-1}\ x\ x\cdots x$〕の列に移動される。すなわち第〔$a_{i-1}\ a_{i-2}\cdots a_0$〕列と第〔$a_0\cdots a_{i-2}\ a_{i-1}$〕列を入れ換える処理が要る。このためには順に

①第〔$a_{i-1}\ a_{i-2}\cdots a_0$〕列と第〔$a_0\ a_{i-2}\cdots a_1\ a_{i-1}$〕列とを入れ換える。

②第〔$a_0\ a_{i-2}\cdots a_1\ a_{i-1}$〕列と第〔$a_0\ a_1\ a_{i-3}\cdots a_2\ a_{i-2}\ a_{i-1}$〕列とを入れ換える。

③以下同様

と、順に上位ビットと下位ビットを交換した列の入れ換えを行えばよい。この時、最下位ビットを第0ビット目として、最下位から第mビット目と第nビット目を交換した列の入れ換えは、その第mビット目と第nビット目がどちらも0あるいはどちらも1でない値の列について2^m-2^n（m

＞n）だけ離れた列間で入れ換えを行えばよい。列方向について、i／2回（iが奇数の時は（i－1）／2回）の列間の入れ換えにより列方向の処理は終了する。行方向についてもまったく同様に扱える。すなわち、これにより一般のデータに対しても本発明におけるビットリバース処理が使用できることがわかった。

なお、上記実施例では、高速フーリエ変換を行うデータ数と並列データ処理装置のＰＥ数が等しい場合を扱った。しかしながら本発明における並列データ処理装置は、必ずしもこの事を限定するものではない。つまり、並列データ処理装置のＰＥ数よりも、処理対象であるデータ数が少なくても、また多くても同様に扱うことができる。

並列データ処理装置のＰＥ数よりもデータ数が少ない場合には、本発明と同様に１ＰＥに１データを対応させ、対応しないＰＥは制御フラグにより演算を行わないことにより、本発明と同様に扱える。

並列データ処理装置のＰＥ数よりもデータ数が多い場合には、次に示す様に１ＰＥに複数のデータを対応させる。例として、４×４個のＰＥで６４ポイントデータを扱う場合を考察する。

第２７図にこの例を示す。６４ポイントをｆ（０）からｆ（６３）で表わす。１６ＰＥで６４ポイントデータを扱うため、１ＰＥには４ポイントのデータを格納する。すなわち、全体で４枚のデータプレーンが存在する。第２７図（ａ）にこの４枚のプレーンを示す。第２７図（ｂ）は４枚のプレーンを広げたものであり、この広げたプレーン上、すなわち仮想的な８×８ＰＥ上で本発明と同様に６４ポイントのデータを格納する。このプレーンの左上４２が第１プレーン，左下４３が第２プレーン，右上４４が第３プレーン，右下４５が第４プレーンに対応する。ＰＥ単位に眺めると、例えば左上端のＰＥ（０，０）には、ｆ（０），ｆ（４），ｆ（３２），ｆ（３６）の４個のデータが格納される。処理は各プレーンごとに本発明における方式を使用することにより同様に扱うことができる。

〔発明の効果〕

以上のように、この発明によれば複数個の同一型の基本演算要素（ＰＥ）を２次元状に相互に接続し、これらのＰＥを行単位あるいは列単位に同時に動作させ、並列に転送，演算等のデータ処理を行うことにより、高速フーリエ変換を高速に実行することができる並列データ処理装置が得られるという効果がある。

また、この発明の別発明によれば、上記２次元アレイを用いて高速フーリエ変換のビットリバース処理も合わせて行うようにしたので、該変換を更に高速に実行することができる並列データ処理装置が得られるという効果がある。

４．図面の簡単な説明

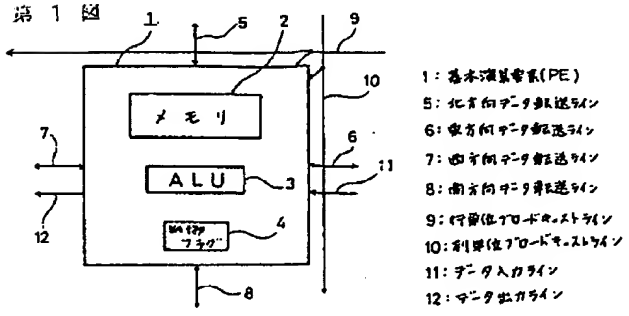
第１図は、この発明の一実施例による基本演算要素（ＰＥ）の内部構成図、第２図はこの発明の一実施例による４×４ＰＥから構成される並列データ処理装置例を示す図、第３図は第４図の高速フーリエ変換例におけるバタフライ演算の記法図、第４図は１６ポイント高速フーリエ変換のアルゴリズム例を示す図、第５図は１６×１６ＰＥから成る並列データ処理装置を構成する各ＰＥの表記法を示す図、第６図は並列データ処理装置へのデータ入力方法を示す図、第７図は並列データ処理装置上での列単位のデータシフト法を示す図、第８図は第７図（ｂ）の処理後の具体的なデータ配置を示す図、第９図は第７図（ｃ）の処理後の具体的なデータ配置を示す図、第１０図は高速フーリエ変換の第１，第２，第３，第４ステージ各々でのデータ分割の方法を示す図、第１１図は同じく第５，第６，第７，第８ステージ各々でのデータ分割の方法を示す図、第１２図は並列データ処理装置上での行単位データシフト法を示す図、第１３図は第１２図（ａ）の処理後の具体的なデータ配置を示す図、第１４図は第１２図（ｂ）の処理後の具体的なデータ配置を示す図、第１５図はビットリバースの例を示す図表、第１６図は並列データ処理装置上のバタフライ演算後のデータ配置を示す図、第１７図は並列データ処理装置上の出力時のデータ配置を示す図、第１８図は行方向

データのビットリバース処理法を示す図、第１９図はデータインデックスの値（a_0，a_1）による処理の相違を示す図表、第２０図は第１９図の具体例を示す図表、第２１図は並列データ処理装置上でのビットリバース処理における列単位のデータシフト方法を示す図、第２２図はデータインデックスの値（a_1，a_2）によるデータの入れ換えの具体例を示す図表、第２３図は並列データ処理装置上でのビットリバース処理における行単位のデータシフト方法を示す図、第２４図は１６×１６ＰＥから成る並列データ処理装置上での２５６個のデータの入力例を各データのインデックス値で示した図、第２５図は第２４図の例においてバタフライ演算が終了した後のデータのインデックスを示した図、第２６図は第２５図のデータのビットリバース処理後のデータの並びを示した図、第２７図は並列データ処理装置のＰＥ数よりもデータの数が多い場合の各ＰＥへのデータ割付け方法を示す図、第２８図は従来の高速フーリエ変換におけるバタフライ演算を行うための演算器の構成図である。１は基本演算要素（ＰＥ）、２はメモリ（記憶手段）、３はＡＬＵ（演算手段）、４は制御フラグ（実行制御手段）、５，６，７，８はデータ転送ライン、９，１０はブロードキャストライン、１１はデータ入力ライン、１２はデータ出力ライン、１３ａは演算アレイ、１３ｂは制御手段。

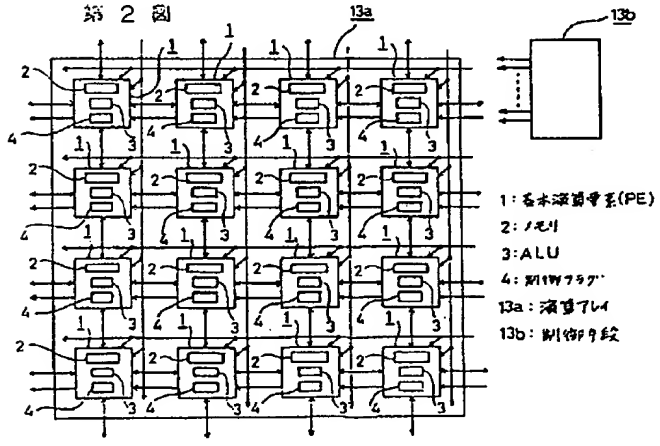
なお、図中、同一符号は同一、又は相当部分を示す。

出願人 工業技術院長

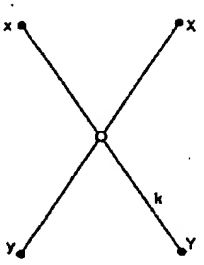
第１図



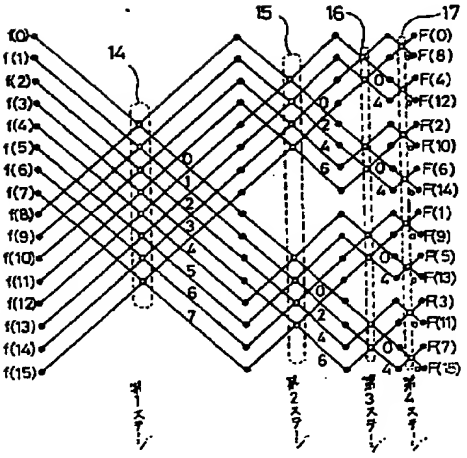
第２図



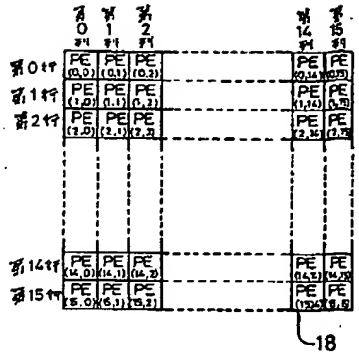
第３図



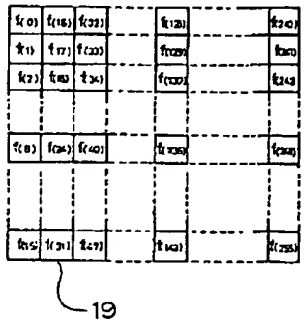
第４図



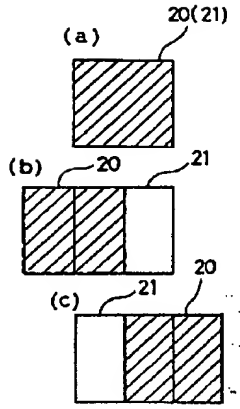
第 5 図



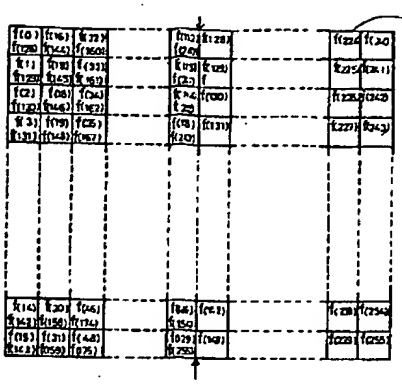
第 6 図



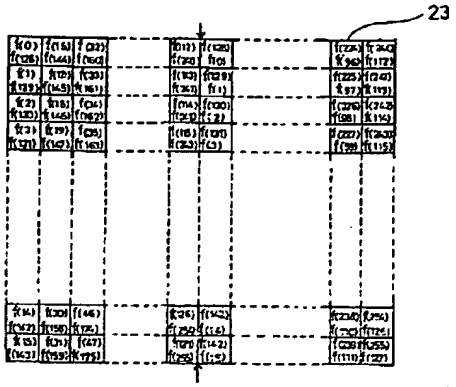
第 7 図



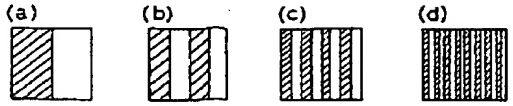
第 8 図



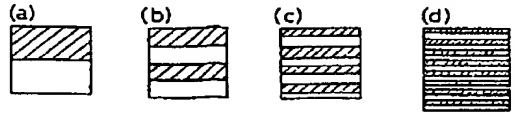
第 9 図



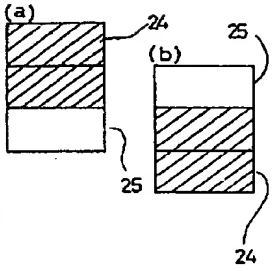
第 10 図



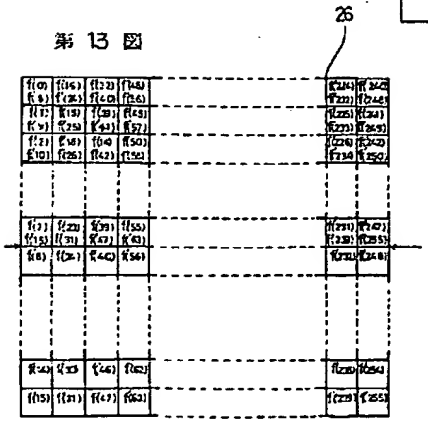
第 11 図



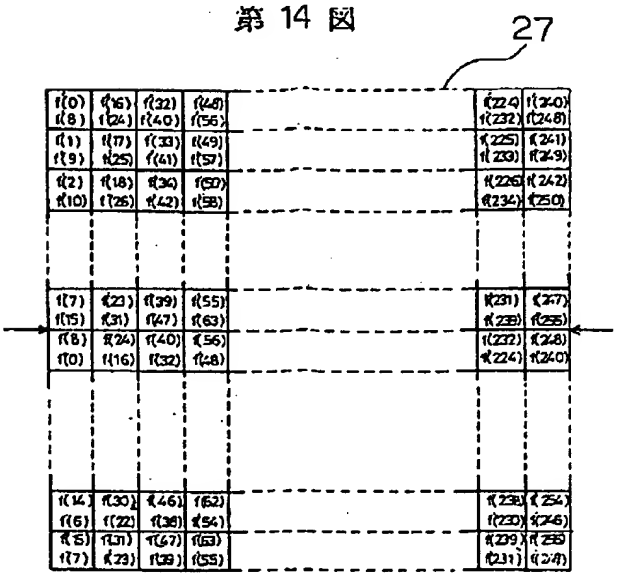
第 12 図



第 13 図



第 14 図

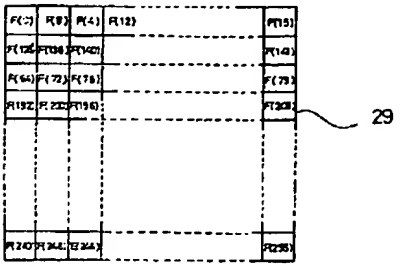


第 15 図

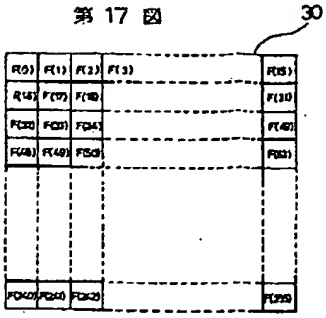
	2進数	2 進 数	ビット リバース
f(0)	(0000)	(0000)	F(0)
f(1)	(0001)	(1000)	F(8)
f(2)	(0010)	(0100)	F(4)
f(3)	(0011)	(1100)	F(12)
f(4)	(0100)	(0010)	F(2)
f(5)	(0101)	(1010)	F(10)
f(6)	(0110)	(0110)	F(6)
f(7)	(0111)	(1110)	F(14)
f(8)	(1000)	(0001)	F(1)
f(9)	(1001)	(1001)	F(9)
f(10)	(1010)	(0101)	F(5)
f(11)	(1011)	(1101)	F(13)
f(12)	(1100)	(0011)	F(3)
f(13)	(1101)	(1011)	F(11)
f(14)	(1110)	(0111)	F(7)
f(15)	(1111)	(1111)	F(15)

28

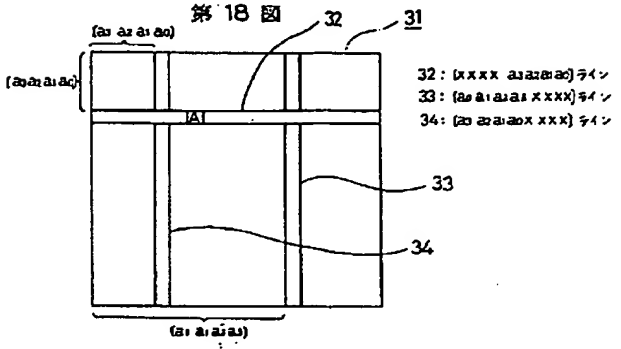
第 16 図



第 17 図



第 18 図



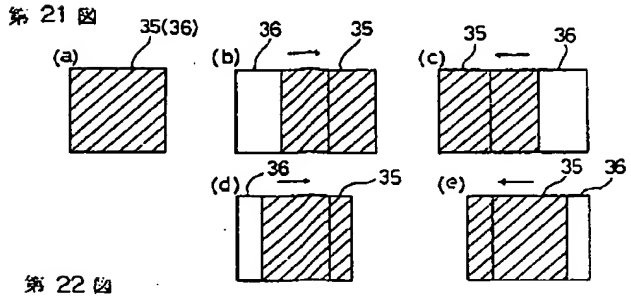
第 19 図

a0	a1	処　理
0	0	入れ換えなし
0	1	互いに7列離れた PEどうしを入れ換える。
1	0	
1	1	入れ換えなし

第 20 図

a3	a2	a1	a0	処　理
0	0	0	0	入れ換えなし
0	0	0	1	
0	0	1	0	入れ換えなし
0	0	1	1	
0	1	0	0	入れ換えなし
0	1	0	1	
0	1	1	0	入れ換えなし
0	1	1	1	
1	0	0	0	
1	0	0	1	入れ換えなし
1	0	1	0	
1	0	1	1	入れ換えなし
1	1	0	0	
1	1	0	1	入れ換えなし
1	1	1	0	
1	1	1	1	入れ換えなし

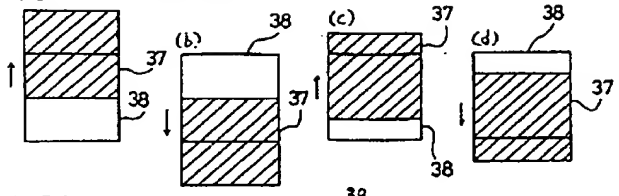
第 21 図



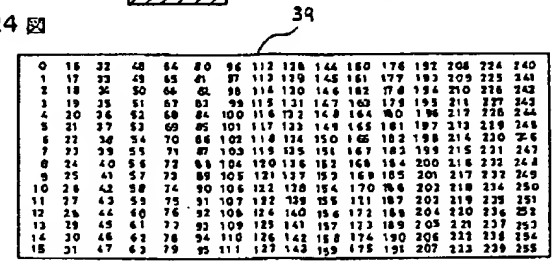
第 22 図

a0 a1 a2 a3	処　理
0 0 0 0	入れ換えなし
0 0 0 1	〃
0 0 1 0	
0 0 1 1	
0 1 0 0	
0 1 0 1	
0 1 1 0	入れ換えなし
0 1 1 1	〃
1 0 0 0	〃
1 0 0 1	〃
1 0 1 0	
1 0 1 1	
1 1 0 0	
1 1 0 1	
1 1 1 0	入れ換えなし
1 1 1 1	〃

第23図

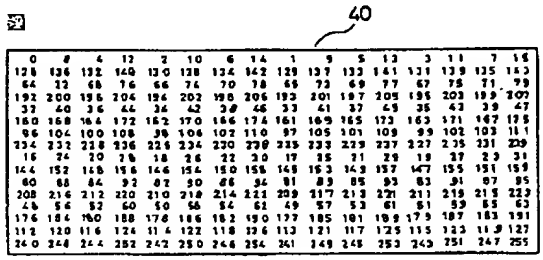


第24図



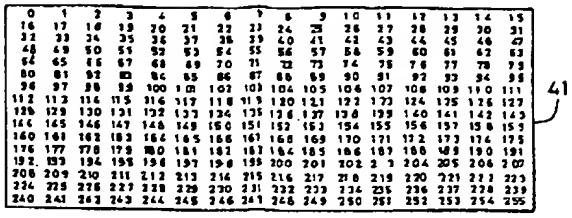
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
11	27	43	59	75	91	107	122	139	155	171	187	203	219	235	251
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

第25図



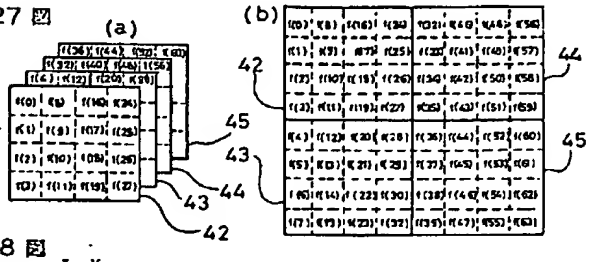
128	136	132	140	130	138	134	142	129	137	133	141	131	139	135	143
64	72	68	76	66	74	70	78	65	73	69	77	67	75	71	79
192	200	196	204	194	202	198	206	193	201	197	205	195	203	199	207
32	40	36	44	34	42	38	46	33	41	37	45	35	43	39	47
160	168	164	172	162	170	166	174	161	169	165	173	163	171	167	175
96	104	100	108	98	106	102	110	97	105	101	109	99	102	103	111
224	232	228	236	226	234	230	238	225	233	229	237	227	235	231	239
16	24	20	28	18	26	22	30	17	25	21	29	19	27	23	31
144	152	148	156	146	154	150	158	145	153	149	157	147	155	151	159
80	88	84	92	82	90	86	94	81	89	85	93	83	91	87	95
208	216	212	220	210	218	214	222	209	217	213	221	211	219	215	223
48	56	52	60	50	58	54	62	49	57	53	61	51	59	55	63
176	184	180	188	178	186	182	190	177	185	181	189	179	187	183	191
112	120	116	124	114	122	118	126	113	121	117	125	115	123	119	127
240	248	244	252	242	250	246	254	241	249	245	253	243	251	247	255

第26図



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

第27図



第28図